UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/720,506	11/24/2003	Mitica Manu	MSFT-2792/306045	4588

41505 7590 08/01/2008

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
08/01/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/720,506	MANU, MITICA	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 April 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4,8,9 and 11-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4,8,9 and 11-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on April 30, 2008.

2. **Claims 1, 2, 4, 8, 9, and 11-22** are pending.

3. **Claims 8, 9, and 18** have been amended.

4. **Claims 3, 5-7, and 10** have been cancelled.

5. The objections to Claims 9, 10, and 18 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claim.

6. The 35 U.S.C. § 112, first paragraph, rejections of Claims 1, 2, 4, 8, 9, and 11-22 as failing to comply with the written description requirement are withdrawn in view of Applicant's arguments and further consideration of paragraph [0038] of the specification. The 35 U.S.C. § 112, first paragraph, rejection of Claim 10 as failing to comply with the written description requirement is withdrawn in view of Applicant's cancellation of the claim.

7. Applicant has failed to address the 35 U.S.C. § 112, first paragraph, rejections of Claims 1, 2, 8, 9, and 12-22 as failing to comply with the enablement requirement. Accordingly, these rejections are maintained and further explained below. The 35 U.S.C. § 112, first paragraph, rejection of Claim 10 as failing to comply with the enablement requirement is withdrawn in view of Applicant's cancellation of the claim.

8. Applicant has failed to address the 35 U.S.C. § 112, second paragraph, rejections of Claims 1, 2, 8, 9, and 12-22 as being incomplete for omitting essential steps. Accordingly, these rejections are maintained and further explained below. The 35 U.S.C. § 112, second paragraph, rejection of Claim 10 as being incomplete for omitting essential steps is withdrawn in view of Applicant's cancellation of the claim.

9. Applicant has failed to address the 35 U.S.C. § 112, second paragraph, rejections of Claims 1, 4, 11, 12, 16, and 20-22 as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Accordingly, these rejections are maintained and further explained below. The 35 U.S.C. § 112, second paragraph, rejection of Claim 10 as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention is withdrawn in view of Applicant's cancellation of the claim. The 35 U.S.C. § 112, second paragraph, rejection of Claim 18 as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention is withdrawn in view of Applicant's amendments to the claim.

10. Applicant has failed to address the 35 U.S.C. § 103(a) rejections of Claims 13 and 14 as being unpatentable over US 6,502,239 (hereinafter "Zgarba") in view of US 6,684,385 (hereinafter "Bailey"). Accordingly, these rejections are maintained and further explained below.

Response to Amendment

Claim Objections

11. **Claims 8 and 11** are objected to because of the following informalities:

- **Claim 8** contains a typographical error: The word "and" after the "specifying a structure of the block …" limitation should be deleted.
- **Claim 11** contains a typographical error: Claim 11 should depend on Claim 8, not Claim 10.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

12. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

13. **Claims 1, 2, 8, 9, and 12-22** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claims contain subject matter, which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

The claimed invention omits the critical step of "generating procedural-oriented output source code from the functional software model" disclosed to be essential to the invention. The element/step is necessary and must occur in the system/method for the claimed invention to function as intended as described in the specification and/or in the preamble of the claims. A claim which omits matter disclosed to be essential to the invention as described in the specification or in other statements of record may be rejected under 35 U.S.C. 112, first paragraph, as not enabling. *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976). See also MPEP § 2164.08(c). Such essential matter may include missing elements, steps or necessary structural cooperative relationships of elements described by the Applicant as necessary to practice the invention.

14. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

15. **Claims 1, 2, 8, 9, and 12-22** are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential steps, such omission amounting to a gap between the steps. See MPEP § 2172.01. The omitted step is: generating procedural-oriented output source code from the functional software model. The omitted step is considered to be critical to the claimed invention, since the element/step is necessary and must occur in the system/method for the claimed invention to function as intended as described in the specification and/or in the preamble of the claims.

16. **Claims 1, 4, 8, 11, 12, 16, and 20-22** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 4, 8, and 11 recite the limitations "at least one of a plurality of programming languages," "the at least one programming language," "at least one target language," and "the at least one target language," respectively. These claims are rendered indefinite because the generated procedural-oriented output source code must be written in a procedural-oriented programming language. In the interest of compact prosecution, the Examiner subsequently interprets these limitations as reading "at least one of a plurality of procedural-oriented programming languages," "the at least one procedural-oriented programming language," "at least

one procedural-oriented target language," and "the at least one procedural-oriented target language," respectively, for the purpose of further examination.

Claim 4 recites the limitation "each of the at least one programming languages." This is awkward claim language and thus, renders the claim indefinite. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "the at least one procedural-oriented programming language" for the purpose of further examination.

Claims 12, 16, and 20 recite limitations relating to features of the object-oriented programming paradigm (*e.g.*, object, class, etc.). These claims are rendered indefinite because a procedural-oriented programming paradigm does not include these features. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to these limitations for the purpose of further examination.

Claim 21 depends on Claim 20 and, therefore, suffers the same deficiency as Claim 20.

Claim 22 recites the limitation "[a] computer-readable storage medium including computer-readable instructions." The claim is rendered indefinite because computer-readable instructions can only be stored or recorded on a computer-readable medium. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "[a] computer-readable storage medium storing computer-readable instructions" for the purpose of further examination.

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. **Claims 1, 2, 4, 8, 9, 11, 12, and 15-21** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zgarba** in view of **US 6,199,195 (hereinafter "Goodwin")**.

As per **Claim 1**, Zgarba discloses:

- a modeler for defining at least one of a plurality of code elements and a structure of a code block and generating a graphical representation of the at least one code element and structure of the code block, wherein the modeler processes input comprising a code block of object-oriented source code and generates from the input a code model comprising a graphical representation of a structure and flow of the code block *(see Column 3: 7-12, "The primary example used to clarify the operation of the embodiment and represented in FIGS. 5-11 Is based around the C++ programming language, although there is no reason the invention could not operate on other object oriented languages or other types of languages, as will become apparent." and 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling*

between components." and 49-67 to Column 4: 1-3, "For example, if the source code language were C++, the software model might provide for the modeling of C++ classes, structs, unions and enumeration classes with similar structures ..."; Column 5: 1-8, "Reverse engineering according to this embodiment is effected by parsing the code and transforming all the elements of the code which can be represented by the software model into a generic meta-model ..." It is inherent that the source code contains various code blocks.).

However, Zgarba does not disclose:

- a computer display;
- procedural-oriented source code;
- wherein the modeler processes input comprising a code block of procedural-oriented source code from an innermost element to an outermost element; and
- a selector for selecting at least one of a plurality of procedural-oriented programming languages in which to generate procedural-oriented output source code from the functional model.

Official Notice is taken that it is old and well-known within the computing art to write the source code in a procedural-oriented programming language. Zgarba discloses that the source code can be written in other types of languages *(see Column 3: 7-12)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to write the source code in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Official Notice is also taken that it is old and well-known within the computing art to process source code from an innermost element to an outmost element. When source code is parsed, it is either parsed from the innermost element to the outermost element or from the outermost element to the innermost element. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include wherein the modeler processes input comprising a code block of object-oriented source code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Goodwin discloses:

- a computer display *(see Figure 1: 110)*; and
- a selector for selecting at least one of a plurality of object-oriented programming languages in which to generate object-oriented output source code from the functional model *(see Column 13: 52-55, "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include a computer display; and a selector for selecting at least one of a plurality of object-oriented programming languages in which to generate object-oriented output source code from the functional model. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a different programming language from the software model.

Official Notice is also taken that it is old and well-known within the computing art to generate procedural-oriented source code from a model. Goodwin discloses that the code generator can support the creation of various files in different programming languages based on user selections *(see Column 13: 52-55)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to generate the file in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Zgarba further discloses:

- a user interface for receiving the definition of the at least one code element and the structure of the code block *(see Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; however, Zgarba does not disclose:

- a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming language and generating procedural-oriented output source code in the at least one procedural-oriented programming language.

Goodwin discloses:

- a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming language and generating procedural-oriented output source code in the at least one procedural-oriented programming language *(see Column 13: 45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include a code generator for receiving the graphical representation of the at least one code element and the structure of the code block and the at least one procedural-oriented programming language and generating procedural-oriented output source code in the at least one procedural-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 8**, Zgarba discloses:

- processing a block of object-oriented programming code and generating from the processed block of object-oriented programming code a functional software model *(see Column 3: 7-12, "The primary example used to clarify the operation of the embodiment and represented in FIGS. 5-11 Is based around the C++ programming language, although there is no reason the*

invention could not operate on other object oriented languages or other types of languages, as will become apparent."; Column 5: 1-8, "Reverse engineering according to this embodiment is effected by parsing the code and transforming all the elements of the code which can be represented by the software model into a generic meta-model ..." It is inherent that the source code contains various code blocks.);

- defining a plurality of code elements within the block of object-oriented programming code *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*;

- specifying a structure of the block of object-oriented programming code including the plurality of code elements *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*; and

- generating from the plurality of code elements and the structure of the block of object-oriented programming code including the plurality of code elements a graphical representation of the plurality of code elements and flow of the block of object-oriented programming code *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components." and 49-67 to Column 4: 1-3, "For example, if the source code language were C++, the software model might provide for the modeling of C++ classes, structs, unions and enumeration classes with similar structures ...")*.

However, Zgarba does not disclose:

- procedural-oriented programming code;

- processing a block of procedural-oriented programming code from an innermost element to an outermost element; and

- specifying at least one target language in which procedural-oriented output source code for the graphical representation is to be generated, wherein the at least one target language specified is different from a language of the processed block of the procedural-oriented programming code.

Official Notice is taken that it is old and well-known within the computing art to write the source code in a procedural-oriented programming language. Zgarba discloses that the source code can be written in other types of languages *(see Column 3: 7-12)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to write the source code in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Official Notice is also taken that it is old and well-known within the computing art to process source code from an innermost element to an outmost element. When source code is parsed, it is either parsed from the innermost element to the outermost element or from the outermost element to the innermost element. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include processing a block of procedural-oriented programming code from an innermost element to an outermost element. The

modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Goodwin discloses:

- specifying at least one target language in which object-oriented output source code for the graphical representation is to be generated, wherein the at least one target language specified is different from a language of the processed block of the object-oriented programming code *(see Column 13: 52-55, "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include specifying at least one target language in which procedural-oriented output source code for the graphical representation is to be generated, wherein the at least one target language specified is different from a language of the processed block of the procedural-oriented programming code. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a different programming language from the software model.

Official Notice is also taken that it is old and well-known within the computing art to generate procedural-oriented source code from a model. Goodwin discloses that the code generator can support the creation of various files in different programming languages based on user selections *(see Column 13: 52-55)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to generate the file in a procedural-

oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 9**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- receiving the definition of the plurality of code elements with the block of procedural-oriented programming code and specifying the structure of the block of procedural-oriented programming code via a user interface *(see Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.").*

As per **Claim 11**, the rejection of **Claim 8** is incorporated; however, Zgarba does not disclose:

- generating the procedural-oriented output source code in the at least one procedural-oriented target language.

Goodwin discloses:

- generating the procedural-oriented output source code in the at least one procedural-oriented target language *(see Column 13: 45-55, "Thus, when the code generator 330 is instantiated (Block, 500, FIG. 5) a selection (Block 502, FIG. 5) is made as to whether a local (IDL) data model source is being used or a schema server data model is being used." and "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include generating the procedural-oriented output source code in the at least one procedural-oriented target language. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises a variable, comment, constant, object, function, method, prototype, member, data type, callback, delegate, reference, field, variant, property, interface, class, type, enumeration, structure, primitive, array, or event handle *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 15**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises an evaluation entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 16**, the rejection of **Claim 15** is incorporated; and Zgarba further discloses:

- wherein the evaluation entity comprises one of a method call, a plurality of code entities, a plurality of code relations, or an instantiation of a class *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 17**, the rejection of **Claim 8** is incorporated; however, Zgarba does not disclose:

- wherein one of the plurality of code elements comprises a passive entity.

Goodwin discloses:

- wherein one of the plurality of code elements comprises a passive entity *(see Column 8: 49-53, "Also shown are a plurality of model adapters 310 for defining a translation of the logical models of the modeling tools 302, 304, 406 into unified models, expressed in a unified modeling language, such as Unified Modeling Language (UML).").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein one of the plurality of code elements comprises a passive entity. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 18**, the rejection of **Claim 17** is incorporated; however, Zgarba does not disclose:

- wherein the passive entity comprises a comment or a modeling diagram.

Goodwin discloses:

- wherein the passive entity comprises a comment or a modeling diagram *(see Column 8: 49-53, "Also shown are a plurality of model adapters 310 for defining a translation of the logical models of the modeling tools 302, 304, 406 into unified models, expressed in a unified modeling language, such as Unified Modeling Language (UML).")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein the passive entity comprises a comment or a modeling diagram. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 19**, the rejection of **Claim 8** is incorporated; and Zgarba further discloses:

- wherein one of the plurality of code elements comprises a block entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.")*.

As per **Claim 20**, the rejection of **Claim 19** is incorporated; and Zgarba further discloses:

- wherein the block entity comprises a method entity, a member entity, a class entity, a namespace entity, or a file entity *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code, and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components.").*

As per **Claim 21**, the rejection of **Claim 20** is incorporated; however, Zgarba does not disclose:

- wherein a many-to-many relationship exists between block entities.

Goodwin discloses:

- wherein a many-to-many relationship exists between block entities *(see Column 4: 31-36, "A 'relationship' defines a link between two object classes." and "Relationships can be one-to-one, one-to-many, or many-to-many.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Goodwin into the teaching of Zgarba to include wherein a many-to-many relationship exists between block entities. The modification would be obvious because one of ordinary skill in the art would be motivated to link classes to other classes.

19. **Claim 22** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Zgarba**.

As per **Claim 22**, Zgarba discloses:

- processing a block of object-oriented programming code and generating from the processed block of object-oriented programming code a functional software model *(see Column 3: 7-12, "The primary example used to clarify the operation of the embodiment and represented in FIGS. 5-11 Is based around the C++ programming language, although there is no reason the invention could not operate on other object oriented languages or other types of languages, as will become apparent."; Column 5: 1-8, "Reverse engineering according to this embodiment is effected by parsing the code and transforming all the elements of the code which can be represented by the software model into a generic meta-model ..." It is inherent that the source code contains various code blocks.)*;

- defining a plurality of code elements within the block of object-oriented programming code *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*;

- specifying a structure of the block of object-oriented programming code including the plurality of code elements *(see Figure 3A; Column 3: 30-33, "For example, code exported from a Sybase. PowerBuilder application shown in FIG. 3A includes information concerning the width, height and position of a window type class w_sort.")*; and

- generating from the plurality of code elements and the structure of the block of object-oriented programming code including the plurality of code elements a graphical representation of the plurality of code elements and flow of the block of object-oriented programming code comprising the functional software model *(see Column 3: 22-26, "The software model 2 does not need to provide all the concepts of the language of the source code,*

and will normally provide modeling for the higher level concepts in the source code. such as classes, attributes, operation/methods etc., or data flow modeling between components." and 49-67 to Column 4: 1-3, "For example, if the source code language were C++, the software model might provide for the modeling of C++ classes, structs, unions and enumeration classes with similar structures ...").

However, Zgarba does not disclose:

- procedural-oriented programming code; and
- processing a block of procedural-oriented programming code from an innermost element to an outermost element.

Official Notice is taken that it is old and well-known within the computing art to write the source code in a procedural-oriented programming language. Zgarba discloses that the source code can be written in other types of languages *(see Column 3: 7-12)*. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to write the source code in a procedural-oriented programming language instead of an object-oriented programming language. The modification would be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Official Notice is also taken that it is old and well-known within the computing art to process source code from an innermost element to an outmost element. When source code is parsed, it is either parsed from the innermost element to the outermost element or from the outermost element to the innermost element. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include processing a block of

procedural-oriented programming code from an innermost element to an outermost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

20. **Claims 13 and 14** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Zgarba** in view of **Goodwin** as applied to Claim 8 above, and further in view of **Bailey**.

As per **Claim 13**, the rejection of **Claim 8** is incorporated; however, Zgarba and Goodwin do not disclose:

- wherein one of the plurality of code elements comprises a code relation.

Bailey discloses:

- wherein one of the plurality of code elements comprises a code relation *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Zgarba to include wherein one of the plurality of code elements comprises a code relation. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; however, Zgarba and Goodwin do not disclose:

- wherein the code relation comprises a mathematical operator.

Bailey discloses:

- wherein the code relation comprises a mathematical operator *(see Column 8: 30-36, "... each program object typically performs some useful function, such as a Boolean operation (e.g., AND, OR, etc.), a mathematical operation, a data acquisition operation ..., renders some comparison (e.g., less than, greater than, equal to, etc.), and so on.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Zgarba to include wherein the code relation comprises a mathematical operator. The modification would be obvious because one of ordinary skill in the art would be motivated to model all aspects of a software program.

Response to Arguments

21. Applicant's arguments filed on April 30, 2008 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a) Applicant respectfully submits that the facts of which the examiner has taken official notice are not so well known as to be capable of instant and unquestioning demonstration as being well known, as is evidenced by the lack of any such instant and unquestioning

demonstration in the evidence of record. Thus, Applicant requests either evidentiary support for the aforementioned features for which official notice was taken, or an affidavit/declaration under 37 C.F.R. 1.104(d)(2) if the official notice is based on personal knowledge of the examiner that isn't supported by such evidentiary support.

Examiner's response:

a) Examiner cites Bailey, US 6,651,246 (hereinafter "Archambault"), and US 7,032,210 (hereinafter "Alloing") as concrete evidence to support the Examiner's taking of Official Notice. The newly added references are added only as directly corresponding evidence to support the prior common knowledge finding, and they do not result in new issues or constitute new grounds of rejection.

Bailey discloses:

- writing the source code in a procedural-oriented programming language *(see Column 1: 38-44, "To generate a software program that can be executed or run by a computer, a software developer or programmer typically chooses a programming language, such as BASIC (Beginner's All-purpose Symbolic Instruction Code), Fortran, C, etc., and writes source code using the keywords, syntax, variable names, data structures, etc. defined by the selected programming language.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Bailey into the teaching of Zgarba to include writing the source code in a procedural-oriented programming language. The modification would

be obvious because one of ordinary skill in the art would be motivated to generate a software model from a block of procedural-oriented source code.

Archambault discloses:

- processing source code from an innermost element to an outmost element *(see Column 6: 38-41, "As indicated in FIG. 1, the loop allocation of the preferred embodiment creates PDG 14 from nested source code 10. PDG builder 12 starts with the innermost nested loop and moves outwards.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Archambault into the teaching of Zgarba to include processing source code from an innermost element to an outmost element. The modification would be obvious because one of ordinary skill in the art would be motivated to process all the code elements of the source code.

Alloing discloses:

- generating procedural-oriented source code from a model *(see Column 1: 33-43, "For example, such application development tools automate the development of business applications by generating application components from existing database definitions. These definitions populate an Information Model which allow the user to formalise high-level specifications via its dedicated entities and to produce a complete operational client/server application. This software application is produced in the form of source code, that is, a group of statements written in a programming language. Examples of programming languages are, C, C++, Java™, etc.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Alloing into the teaching of Zgarba to include

generating procedural-oriented source code from a model. The modification would be obvious because one of ordinary skill in the art would be motivated to regenerate source code in a procedural-oriented programming language from the software model.

In the Remarks, Applicant argues:

b) When explaining how and why the examiner can not change the principle of operation of the cited reference or render the cited reference inoperable for its intended purpose. See MPEP 2143.01 V and VI. As described herein, the reverse-engineering/forward-engineering methods of Zgarba would not operate as described if an initial language was different from a target language, as recited in the claims.

Examiner's response:

b) Examiner disagrees. It is noted that the round-trip engineering feature (*i.e.*, keeping a software model in synchronization with the source code it represents) of Zgarba does not require the programming language of the source code represented by the software model to be the same as the programming language of the source code generated from the software model. In other words, in order to keep the software model and the source code in synchronization, the source code generated by reverse engineering and the source code generated by forward engineering should be essentially the same. There is no restriction placed upon the choice of the programming language of the source code. As long as the details of the software project are fully represented in the software model and the source code, the software model and the source code can be kept synchronized.

Thus, the proposed modification or combination of the prior art would not change the principle of operation of the prior art invention being modified. One of ordinary skill in the art would be motivated to modify Zgarba to incorporate the teaching of Goodwin in order to regenerate source code in a different programming language from the software model.

In the Remarks, Applicant argues:

c) Applicant respectfully disagrees that Goodwin teaches the claimed element. Rather, the selection in Goodwin (col 13, lines 44-55) is not the selection of at least one of a plurality of procedural-oriented programming languages for the output source code. Goodwin's selection is a selection of a data model to use for the creation of code. The language itself is not chosen in Goodwin, but rather, the data model to be used for the creation of code.

Examiner's response:

c) Examiner disagrees. As previously pointed out in the Non-Final Rejection (mailed on 01/31/2008), Goodwin clearly discloses "a selector for selecting at least one of a plurality of object-oriented programming languages in which to generate object-oriented output source code from the functional model" *(see Column 13: 52-55, "Thus, the code generator 330 can support the creation of, for example, IDL, JAVA or C++ files (CORBA, Java RMI, and/or COM) based on certain user preferences and selections.")*. Note that the code generator supports generating output files in various programming languages and it generates the output files in a particular programming language based on a user selection.

Conclusion

22. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

23. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/QC/
July 17, 2008

/Wei Zhen/

Supervisory Patent Examiner, Art Unit 2191